НОВЫЕ СРЕДСТВА
КИБЕРНЕТИКИ, ИНФОРМАТИКИ,
ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ
И СИСТЕМНОГО АНАЛИЗА

А.А. БАРКАЛОВ, Л.А. ТИТАРЕНКО, А.Н. МИРОШКИН

УДК 004.3

МОДИФИКАЦИЯ СИСТЕМЫ АДРЕСАЦИИ МИКРОКОМАНД В УСТРОЙСТВЕ УПРАВЛЕНИЯ С РАЗДЕЛЕНИЕМ КОДОВ

Ключевые слова: *граф-схема алгоритма, управляющее устройство, композиционное микропрограммное устройство управления, разделение кодов, программируемые логические интегральные схемы типа FPGA, адресация микрокоманд.*

ВВЕДЕНИЕ

Любая цифровая система состоит из устройства управления (УУ) и операционного автомата. Первое вырабатывает последовательность управляющих сигналов, под воздействием которых операционный автомат выполняет определенные микрооперации [1]. Для задания последовательности действий используется язык граф-схем алгоритма (ГСА). Функционирование УУ, реализующего ГСА, описывается моделью цифрового автомата [2]. Практическая реализация различных типов УУ характеризуется такими показателями, как аппаратурные затраты и временные характеристики (длительность такта синхронизации, наработка до отказа). На практике, как правило, актуальна проблема минимизации аппаратурных затрат [3], решение которой зависит от особенностей алгоритма управления и используемого при реализации устройства элементного базиса. В случае если алгоритм носит линейный характер, целесообразно использовать модель композиционного микропрограммного устройства управления (КМУУ) [4, 5].

Настоящая работа посвящена исследованию процесса реализации схемы КМУУ в базисе программируемых логических интегральных схем типа FPGA (field-programmable gate array) [6, 7]. Функциональным элементом FPGA служит таблица преобразования (Look-Up Table — LUT). Каждый LUT-элемент можно использовать не только в качестве функциональных генераторов, но и как синхронную память небольшой емкости. Кроме LUT-элементов большинство современных микросхем FPGA содержат специальные программируемые блоки памяти, каждый из которых представляет собой синхронное оперативное запоминающее устройство (ОЗУ) емкостью 18 Кбит [8]. Блочное ОЗУ эффективнее применяется в схемах для хранения относительно большого количества данных, чем упомянутая распределенная память на базе LUT-элементов. Реализация конечных автоматов с памятью в базисе FPGA описана в [9–13].

Далее рассмотрены некоторые подходы к модификациям системы адресации микрокоманд, которые основаны на модели КМУУ с разделением кодов [4], однако предложенную идею можно реализовать и для других структур управляющих устройств с памятью.

Исходными данными для синтеза УУ является граф-схема управляющего алгоритма. Пусть произвольная ГСА Г состоит из множества вершин B и дуг E, соединяющих эти вершины. При этом $B = \{b_0, b_E\} \cup B_1 \cup B_2$, где b_0, b_E — начальная и конечная вершины ГСА а, B_1, B_2 — множества операторных и условных вершин соответственно. Вершины $b_m \in B_1$ содержат наборы микроопераций $Y(b_m) \subseteq Y$, где $m = \overline{1, M}$, $M = |B_1|$ — общее количество операторных вершин ГСА, $Y = \{y_1, \ldots, y_N\}$ — множество микроопераций (выходных функций автомата). Вершины $b_q \in B_2$ содержат элементы множества логических условий $X = \{x_1, \ldots, x_L\}$. Введем определения, необходимые для дальнейшего изложения.

Определение 1. Операторной линейной цепью α_g ГСА Г называется конечная последовательность операторных вершин $\langle b_{g_1}, \ldots, b_{g_{F_g}} \rangle$, для любой пары соседних компонентов которой существует дуга $\langle b_{g_i}, b_{g_{i+1}} \rangle \in E$, где $i = \overline{1, F_g - 1}$, F_g — число компонентов в операторных линейных цепях (ОЛЦ) α_g.

Определение 2. Операторная вершина $b_m \in B^g$, где $m = \overline{1, F_g}$, $B^g \subseteq B_1$ — множество операторных вершин, входящих в ОЛЦ α_g, называется ее входом, если существует дуга $\langle b_t, b_m \rangle \in E$, где $b_t \notin B^g$. Каждая ОЛЦ α_g имеет произвольное число входов, образующих множество $I(\alpha_g) = \{I_g^1, I_g^2, \ldots\}$.

Определение 3. Операторная вершина $b_m \in B^g$ называется выходом ОЛЦ α_g, если существует дуга $\langle b_m, b_t \rangle \in E$, где $b_t \notin B^g$. Произвольная ОЛЦ α_g имеет только один выход, обозначаемый O_g.

Определение 4. Цепи α_i и α_j называются псевдоэквивалентными ОЛЦ (ПОЛЦ), если существуют дуги $\langle b_i, b_t \rangle \in E$ и $\langle b_j, b_t \rangle \in E$, где b_i, b_j — выходы ОЛЦ α_i, α_j соответственно.

Определение 5. Граф схема алгоритма Г называется линейной, если выполняется условие

$$\frac{M}{G} \geq 2, \tag{1}$$

где G — число операторных линейных цепей ГСА, т.е ГСА является линейной, если число операторных вершин алгоритма превышает минимальное количество цепей не менее чем в два раза. При выполнении условия (1) для реализации алгоритма управления целесообразно использовать модель КМУУ [4].

Для нахождения разбиения множества B_1 операторных вершин на минимальное количество ОЛЦ используется методика, описанная в [4]. После формирования множества $C = \{\alpha_1, \ldots, \alpha_G\}$ каждая ОЛЦ α_g содержит F_g компонентов и ей присваивается двоичный код $K(\alpha_g)$ разрядности

$$R_1 = \lceil \log_2 G \rceil, \tag{2}$$

а каждому компоненту b_{g_i} — код $K(b_{g_i})$ разрядности

$$R_2 = \lceil \log_2 (F_{\max}) \rceil, \tag{3}$$

где $F_{\max} = \max(F_1, \ldots, F_G)$ — количество компонентов в ОЛЦ максимальной длины. Для кодирования ОЛЦ используются элементы множества τ, а для кодирования компонентов ОЛЦ — элементы множества T, где $|\tau| = R_1$, $|T| = R_2$.

Кодирование компонентов выполняется в естественном порядке, т.е.

$$K(b_{g_{i+1}}) = K(b_{g_i}) + 1, \ (g = \overline{1, G}; \ i = \overline{1, F_g}). \tag{4}$$

Операторная вершина b_m соответствует микрокоманде MI_m, адрес $A(MI_m)$ которой определяется как

$$A(MI_m) = K(\alpha_g) * K(b_{g_i}), \tag{5}$$

где $*$ — знак конкатенации, а вершина b_m соответствует компоненту b_{g_i} ОЛЦ α_g.

Применение принципа разделения кодов приводит к тому, что схема формирования адреса (СФА) реализует систему функций возбуждения счетчика СТ и регистра RG:

$$\begin{aligned} \Phi &= \Phi(\tau, X); \\ \Psi &= \Psi(\tau, X). \end{aligned} \tag{6}$$

Модель КМУУ с разделением кодов (Code Sharing, CS-структура) приведена на рис. 1 и функционирует следующим образом. В начале работы по сигналу Start триггер ТВ разрешения чтения из управляющей памяти (УП) переключается в состояние логической единицы. Схема формирования адреса обеспечивает переход на входную вершину одной из ОЛЦ α_g. Сформированный для нее адрес, попав в счетчик, участвует в выборке определенной микрокоманды из памяти. Если текущая вершина не является выходом ОЛЦ, формируется сигнал y_0, который позволяет счетчику увеличить свое содержимое, тем самым адресуя следующий компонент ОЛЦ α_g согласно правилу естественной адресации микрокоманд (4). По достижению выхода O_g ОЛЦ α_g сигнал y_0 не формируется, что приводит к записи в счетчик адреса перехода на входную вершину очередной ОЛЦ. Этот адрес формируется СФА по коду текущей ОЛЦ, который находится в регистре RG. Функционирование устройства завершается, когда сформирован сигнал y_E, запрещающий дальнейшее считывание микрокоманд. Этот сигнал добавляется во все вершины, выходы которых связаны с конечной вершиной ГСА.

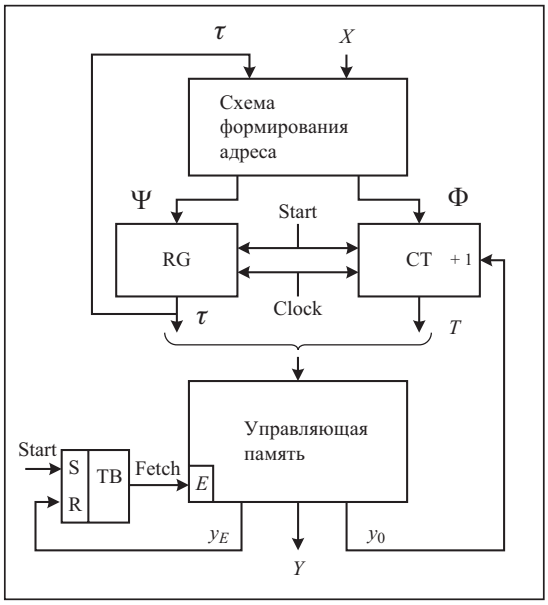

Рис. 1. Структурная схема КМУУ с разделением кодов

При реализации схемы КМУУ в базисе FPGA для блоков, кроме УП, применяются LUT-элементы и распределенные элементы памяти (триггеры-защелки), а для реализации памяти — встроенные блоки памяти. Если при синтезе КМУУ часть ресурсов встроенных блоков памяти не используется, их можно задействовать для уменьшения числа LUT-элементов в схеме адресации, что снизит аппаратурные затраты при реализации устройства и улучшит временные характеристики полученной схемы.

ОСНОВНАЯ ИДЕЯ ПРЕДЛАГАЕМОГО МЕТОДА

Переходы из псевдоэквивалентных цепей представлены в таблице переходов КМУУ строками, отличающимися лишь источником кода для СФА. Таким образом, их можно заменить одной строкой с кодом класса ПОЛЦ в качестве источника, вследствие чего уменьшится количество строк таблицы, а также

число аргументов и термов в функциях системы (6). Коды классов ПОЛЦ можно хранить в свободных ресурсах встроенных блоков памяти микросхемы FPGA. В настоящей статье предложены два подхода к модификации системы адресации микрокоманд: расширение формата микрокоманд, т.е. включение в формат дополнительного поля, которое будет содержать код класса ПОЛЦ; модификация цепей, т.е. включение в исходную ГСА дополнительной вершины с кодом класса ПОЛЦ в конце каждой операторной линейной цепи, не связанной с конечной вершиной.

Пусть ОЛЦ $\alpha_g \in C_1$, если ее выход O_g не связан с конечной вершиной b_E. Найдем разбиение $\Pi_C = \{B_1, \ldots, B_I\}$ множества C_1 на классы ПОЛЦ и закодируем каждый из них двоичным кодом разрядности $R_3 = \lceil \log_2 I \rceil$. Для кодирования классов ПОЛЦ используем переменные множества $Z = \{z_1, \ldots, z_{R_3}\}$. Тогда система (6) преобразуется к виду

$$\Phi = \Phi(Z, X); \qquad \Psi = \Psi(Z, X). \tag{7}$$

При расширении формата микрокоманд их количество в памяти равно соответствующему значению для КМУУ с разделением кодов

$$M_1 = (G-1) \cdot 2^{\lceil \log_2 |\alpha_{\max}| \rceil} + |\alpha_{\min}|,$$

где $\alpha_{\max}$, $\alpha_{\min}$ — цепи с максимальным и минимальным количеством компонентов соответственно. Для обеспечения минимального числа слов микропрограммы необходимо ОЛЦ $\alpha_{\min}$ присвоить максимальный из используемых кодов ОЛЦ, в результате соответствующее содержимое памяти поступит в область с максимальными адресами.

Разрядность каждого слова после включения дополнительного поля будет равна $N_1 = N_Y + 2 + R_3$, где N_Y — количество разрядов, необходимых для кодирования множества микрокоманд $y_i \in Y$ (при унитарном кодировании микроопераций $N_Y = |Y|$), а константа 2 резервирует два разряда для хранения внутренних переменных: y_0 и y_E. Разрядности регистра и счетчика равны соответствующим значениям (2) и (3).

Модификация ОЛЦ может увеличить количество слов микропрограммы до величины $M_2 = (G-1) \cdot 2^{\lceil \log_2 (|\alpha_{\max}|+1) \rceil} + |\alpha_{\min}| + 1$.

Разрядность слова при тривиальной реализации предложенного подхода определяется как $N_2 = \max(N_Y; R_3) + 2$.

Разрядность регистра и счетчика после модификации цепей определяется формулами (2) и (3). Однако параметр $F_{\max}$ может увеличиться, что приведет к увеличению количества разрядов регистра. Отметим, что при увеличении разрядности счетчика СФА будет по-прежнему формировать R_2 переменных. Это связано с тем, что непосредственный переход к микрокоманде, которая содержит лишь код класса ПОЛЦ, никогда не выполнится. При этом старший разряд данных для счетчика всегда будет равен нулю.

Условимся, что с параметрами M_i и N_i микропрограмму можно реализовать с использованием одного блока встроенной памяти.

Обозначим FCS и MCS представленные на рис. 2, *а*, *б* модели КМУУ соответственно.

Принципы функционирования базовой модели КМУУ с разделением кодов (см. рис. 1) и моделей FCS и MCS аналогичны. Метод синтеза рассмотренных структур включает следующие этапы:

— формирование для ГСА Г множеств C, C_1 и Π_C;

— оптимальное кодирование цепей $\alpha_g \in C_1$ и их компонентов;

— кодирование классов псевдоэквивалентных цепей $B_i \in \Pi_C$;

— формирование содержимого УП;

— формирование таблицы переходов КМУУ и системы функций переходов (7);

— синтез схемы УУ в заданном базисе.

Первый этап выполняется по описанной в [4] методике, при этом количество G ОЛЦ $\alpha_g \in C$ является минимально возможным. Разбиение Π_C формируется на основе определения 4 тривиальным образом.

Оптимальное кодирование ОЛЦ $\alpha_g \in C_1$ выполняется по методике, аналогичной оптимальному кодированию состояний автомата Мура [15]. Для решения этой задачи используются известные методы символьной минимизации [3]. Компоненты всех ОЛЦ кодируются тривиальным способом: первому компоненту присваивается код, десятичный эквивалент которого равен нулю, второму — единице, далее — двум и т.д. Такое кодирование удовлетворяет равенству (4).

Способ кодирования классов $B_i \in \Pi_C$ должен минимизировать количество термов в функции $z_r \in Z$. Адаптировав известный для D-триггеров метод кодирования состояний [15], получим, что чем больше обобщенных интервалов необходимо для представления класса $B_i \in \Pi_C$, тем меньше единиц должен содержать его код.

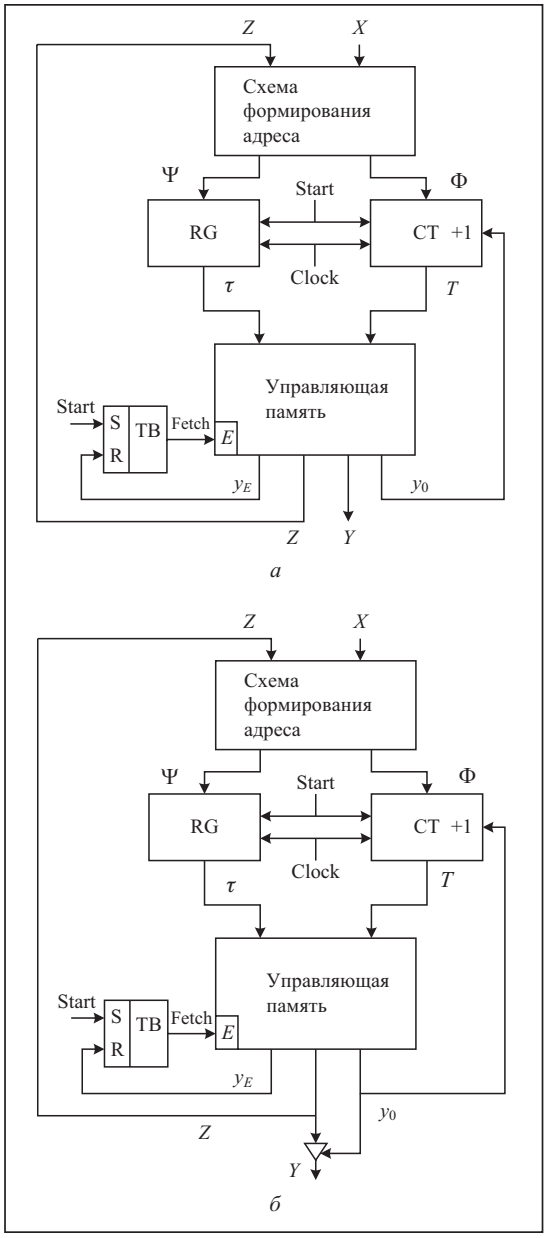

Рис. 2. Структурные схемы КМУУ с модификациями системы адресации микрокоманд: с расширением формата микрокоманд (*а*); с модификацией ГСА (*б*)

Содержимое УП формируется в виде таблицы с полями $A(MI_m)$, $Y(b_m)$, y_0, y_E, $C(B_i)$. Причем для FCS-структуры КМУУ поля $Y(b_m)$ и $C(B_i)$ являются отдельными столбцами наравне с остальными полями. В случае MCS-структуры эти поля записываются в один столбец, содержимое которого трактуется в зависимости от значения переменной y_0. При этом ширина столбца выбирается как $\max(N_Y; R_3)$, а к кодам в поле меньшей ширины добавляются незначащие нули.

Перед построением содержимого памяти происходит преобразование исходной ГСА [4]. Если вершина b_{g_i} ОЛЦ $\alpha_g \in C_1$ не является выходом ОЛЦ, в нее вводится переменная $y_0 = 1$, которая определяет естественную адресацию для следующего компонента ОЛЦ. Если $\alpha_g \notin C_1$, то в вершину O_g вводится переменная $y_E = 1$, которая завершит работу автомата.

Таблица переходов для рассматриваемых КМУУ содержит следующие столбцы: B_i — класс ПОЛЦ, из которого выполняется переход; $k(B_i)$ — код класса ПОЛЦ; b_m — вершина, на которую выполняется переход; $A(MI_m)$ — адрес микрокоманды, соответствующей вершине b_m; X_h — конъюнкция логических переменных, определяющих переход из выхода O_g ОЛЦ $\alpha_g \in B_i$ в вершину b_m в h-й строке таблицы переходов; Ψ_h, Φ_h — функции возбуждения регистра и счетчика соответственно; h — номер терма (строки таблицы переходов).

Каждая функция системы (7) представляется в виде

$$d_i = \bigvee_{h=1}^{H} \left(P_i^h \wedge B_h \wedge \left(\bigwedge_{l=1}^{L} C_l^h x_l^h \right) \right), \quad (i = \overline{1, R}), \tag{8}$$

где d_i — функция возбуждения i-го разряда регистра состояния автомата ($d_i = \tau_i$ при $i = \overline{1, R_2}$ и $d_{i+R_2} = T_i$ при $i = \overline{1, R_1}$); P_i^h — переменная, определяющая наличие функции d_i в строке h таблицы переходов ($P_i^h = 1$, если функция d_i имеется в строке h и $P_i^h = 0$ в противном случае); B_h — конъюнкция разрядов q_r^h кода класса ПОЛЦ (q_r^h — значение r-го разряда ($r = \overline{1, R_3}$) кода $k(B_i)$ класса $B_i \in \Pi_C$ ПОЛЦ в строке h таблицы переходов, $q_r^h = q_r$, если r-й разряд кода равен логической единице и $q_r^h = \bar{q}_r$ в противном случае); x_l^h — переменная, определяющая значение логического условия x_l в строке h таблицы переходов ($x_l^h = x_l$ в случае, если переход в строке h происходит при выполнении условия, и $x_l^h = \bar{x}_l$ — при невыполнении); C_l^h — переменная, определяющая наличие логического условия x_l в строке h таблицы переходов ($C_l^h = 1$ при наличии сигнала x_l и $C_l^h = 0$ при его отсутствии); H — количество строк таблицы переходов; R_3 — разрядность кода класса ПОЛЦ; L — количество логических условий.

Синтез схемы КМУУ сводится к реализации модели управляющего автомата в базисе FPGA с помощью одного из стандартных прикладных пакетов [16, 17] и выходит за рамки нашей статьи.

ПРИМЕР ПРИМЕНЕНИЯ ПРЕДЛАГАЕМОГО МЕТОДА

Пусть некоторая ГСА Γ_1 содержит $M = 17$ операторных вершин, которые образуют множество $C = \{\alpha_0, \ldots, \alpha_7\}$ ОЛЦ, где $\alpha_0 = \langle b_0, b_1, b_2 \rangle$, $\alpha_1 = \langle b_3, b_4, b_5 \rangle$, $\alpha_2 = \langle b_6, b_7 \rangle$, $\alpha_3 = \langle b_8, b_9, b_{10} \rangle$, $\alpha_4 = \langle b_{11}, b_{12} \rangle$, $\alpha_5 = \langle b_{13}, b_{14} \rangle$, $\alpha_6 = \langle b_{15}, b_{16} \rangle$, $\alpha_7 = \langle b_{17}, b_E \rangle$. Получено $G = 8$ ОЛЦ из $M = 17$ операторных вершин. Условие (1) выполняется, следовательно, применение модели КМУУ целесообразно.

Цепь $\alpha_7 \notin C_1$, поскольку она содержит конечную вершину ГСА. Множество логических условий включает $L = 4$ элементов, автомат вырабатывает $N = 6$ управляющих сигналов. Из множества C_1 получено разбиение $\Pi_C = \{B_0, \ldots, B_3\}$ на классы ПОЛЦ, где $B_0 = \{\alpha_0, \alpha_5\}$, $B_1 = \{\alpha_1, \alpha_2, \alpha_4\}$, $B_2 = \{\alpha_3\}$, $B_3 = \{\alpha_6\}$.

Для кодирования вершин ГСА согласно (2) и (3) достаточно $R_1 = 3$ элементов множества $\tau = \{\tau_1, \tau_2, \tau_3\}$ и $R_2 = 2$ элементов множества $T = \{T_1, T_2\}$. Коды вершин ГСА Γ_1 приведены в табл. 1, в скобках даны добавленные вершины для КМУУ с модификацией ОЛЦ. Согласно выражению (5) код вершины определяется как конкатенация кода ОЛЦ и кода самого компонента, т.е. $k(b_0) = 00000$, $k(b_3) = 00100$, $k(b_{16}) = 11001$ и т.д. Закодируем классы $B_i \in \Pi_C$ следующим об-

разом: $k(B_0)=00$, $k(B_1)=01$, $k(B_2)=10$, $k(B_3)=11$. При синтезе КМУУ с модификацией цепей алгоритма управления в конец каждой ОЛЦ $\alpha_g \in C_1$ вводится вершина, содержащая код $k(B_i)$, где $\alpha_g \in B_i$.

Таблица 1

T_1T_2	Коды вершин для ОЛЦ ($\tau_3\tau_2\tau_1$)							
	α_0 (000)	α_1 (001)	α_2 (010)	α_3 (011)	α_4 (100)	α_5 (101)	α_6 (110)	α_7 (111)
00	b_0	b_3	b_6	b_8	b_{11}	b_{13}	b_{15}	b_{17}
01	b_1	b_4	b_7	b_9	b_{12}	b_{14}	b_{16}	—
10	b_2	b_5	(b_{20})	b_{10}	(b_{22})	(b_{23})	(b_{24})	—
11	(b_{18})	(b_{19})	—	(b_{21})	—	—	—	—

Напомним, что переменная y_0 записана в вершинах, которые не являются выходами ОЛЦ $\alpha_g \in C_1$. Переменная y_E записана в вершинах, связанных с конечной вершиной алгоритма. Остальные поля множества Y формируются согласно содержимому ГСА. Фрагменты содержимого памяти для исследуемых структур КМУУ приведены на рис. 3, где жирным шрифтом выделены коды ПОЛЦ, которые участвуют в формировании адреса перехода, а светлым — незадействованные области памяти. Аналогично получается содеримое и для всех остальных цепей $\alpha_g \in C$.

По исходной граф-схеме строится таблица переходов, на основе которой получают функции (7). Рассмотрим таблицу переходов для ГСА Γ_1.

Адрес		Микрокоманда										b_m	ОЛЦ α_i
$\tau_1\ \tau_2\ \tau_3$	$T_2\ T_1$	y_E	y_1	y_2	y_3	y_4	y_5	y_6	y_0	z_2	z_1		
0 0 1	0 0	0	1	0	1	0	0	0	1	0	0	b_3	α_1
	0 1	0	0	0	0	1	0	0	1	0	0	b_4	
	1 0	0	0	0	0	0	1	0	0	**0**	**1**	b_5	
	1 1	0	0	0	0	0	0	0	0	0	0	—	
0 1 0	0 0	0	0	1	0	0	0	0	1	0	0	b_6	α_2
	0 1	0	0	0	1	0	0	1	0	**0**	**1**	b_7	
	1 0	0	0	0	0	0	0	0	0	0	0	—	
	1 1	0	0	0	0	0	0	0	0	0	0	—	

а

Адрес		Микрокоманда								b_m	ОЛЦ α_i
$\tau_1\ \tau_2\ \tau_3$	$T_2\ T_1$	y_E	y_1 (z_2)	y_2 (z_1)	y_3	y_4	y_5	y_6	y_0		
0 0 1	0 0	0	1	0	1	0	0	0	1	b_3	α_1
	0 1	0	0	0	0	1	0	0	1	b_4	
	1 0	0	0	0	0	0	1	0	1	b_5	
	1 1	0	**0**	**1**	0	0	0	0	0	b_{19}	
0 1 0	0 0	0	0	1	0	0	0	0	1	b_6	α_2
	0 1	0	0	0	1	0	0	1	1	b_7	
	1 0	0	**0**	**1**	0	0	0	0	0	b_{20}	
	1 1	0	0	0	0	0	0	0	0	—	

б

Рис. 3. Фрагменты содержимого управляющей памяти для КМУУ: с расширением формата микрокоманд (*а*); с модификацией ОЛЦ алгоритма управления (*б*)

Представим каждую функцию системы (7) в виде функций (8):

$$B_0 = \overline{z_1}\,\overline{z_0};\ B_1 = \overline{z_1} z_0;\ B_2 = z_1\overline{z_0};\ B_3 = z_1 z_0;$$

$$\tau_3 = B_1 \vee B_2;\ \tau_2 = B_0\overline{x_1} \vee B_1;$$

$$\tau_1 = B_0 x_1 \vee B_0\overline{x_1}x_2 \vee B_1\overline{x_4} \vee B_2\overline{x_3} \vee B_3;\quad T_2 = B_3;\ T_1 = 0.$$

Как уже отмечалось, этап реализации схемы КМУУ в данной статье не рассматривается. Однако авторами реализована САПР, позволяющая совместно с пакетом Xilinx ISE Webpack синтезировать предложенные схемы КМУУ.

Таблица 2

B_i	$k(B_i)$	b_m	$A(MI_m)$	X_h	Ψ_h	Φ_h	h
B_0	00	b_3	00100	x_1	τ_1	—	1
		b_8	01100	$\overline{x}_1 x_2$	$\tau_2 \tau_1$	—	2
		b_6	01000	$\overline{x}_1 \overline{x}_2$	τ_2	—	3
B_1	01	b_{15}	11000	x_4	$\tau_3 \tau_2$	—	4
		b_{17}	11100	$\overline{x}_4$	$\tau_3 \tau_2 \tau_1$	—	5
B_2	10	b_{11}	10000	x_3	τ_3	—	6
		b_{13}	10100	$\overline{x}_3$	$\tau_3 \tau_1$	—	7
B_3	11	b_5	00110	1	τ_1	T_2	8

РЕЗУЛЬТАТЫ ИССЛЕДОВАНИЙ

Анализ проведен с помощью разработанной САПР управляющих автоматов, которая по описанию граф-схемы алгоритма управления в формате XML синтезирует модели структур управляющего автомата. Под моделью понимается VHDL-описание схемы устройства и файлы прошивки ПЗУ (для устройств с памятью). Эти модели передаются в систему Xilinx ISE Webpack, где происходит их имплементация в одну из доступных микросхем. Файлы проекта имплементации устройства содержат информацию о задействованных аппаратурных ресурсах микросхемы, о критических путях следования сигнала, временных показателях устройства, потребляемой устройством мощности и др.

Для эксперимента выбраны ГСА со следующими параметрами: от 10 до 500 вершин с шагом 10; от 50 % до 90 % операторных вершин с шагом 10 %; 15 микроопераций; пять логических условий.

Для каждой ГСА синтезирована структура КМУУ с разделением кодов (CS), а также исследуемые FCS- и MCS-структуры. Для сравнения с классом автоматов с жесткой логикой также синтезированы автоматы Мили. Каждое измерение, представленное на графиках, является средним значением результатов синтеза пяти различных ГСА с одинаковыми параметрами.

Анализ аппаратурных затрат показал эффективность применения предлагаемых методик для всех исследуемых ГСА (рис. 4).

Предложенные структуры КМУУ (см. рис. 4) требуют меньше аппаратурных затрат, чем базовая КМУУ с разделением кодов и автомат Мили. Отметим, что с ростом доли операторных вершин в ГСА аппаратурные затраты для реализации автомата Мили увеличиваются, а для реализации устройства класса КМУУ — уменьшаются. Расхождения в значениях аппаратурных затрат для FCS- и MCS-структур КМУУ минимальны и имеют характер статистических отклонений. Для дальнейшего сравнения данных структур исследуем временные характеристики полученных схем.

В качестве временных характеристик выбраны период синхросигнала и длительность формирования выходных функций. Первый параметр имеет смысл, когда логические условия заданы в зависимости от режима работы. Если выбор ветви алгоритма обусловлен результатом предыдущей операции, то имеет смысл второй временной параметр, определяющий максимальное время после формирования последнего из логических условий, через которое устройство сгенерирует выходные функции.

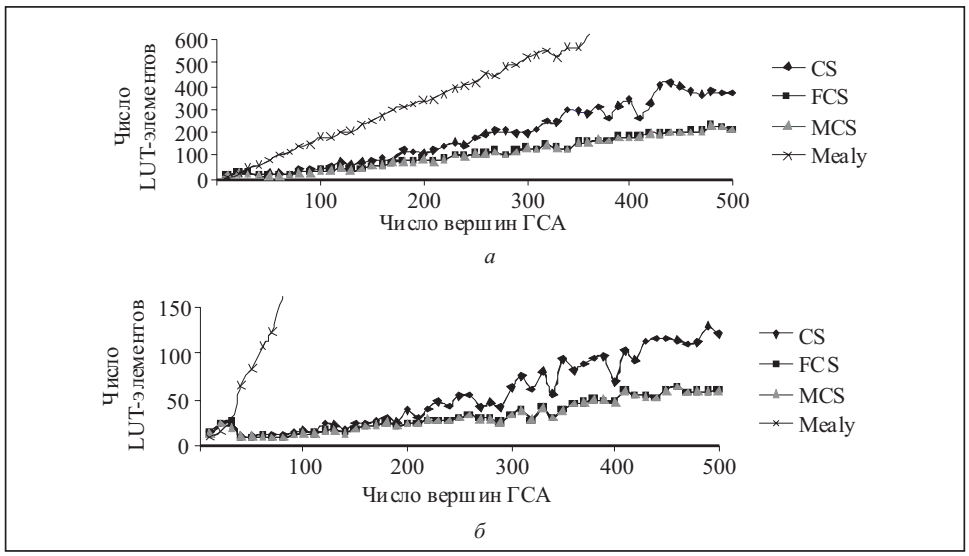

Рис. 4. Графики аппаратурных затрат при реализации различных структур КМУУ, когда ГСА содержит 70 % операторных вершин (*а*) и 90 % операторных вершин (*б*)

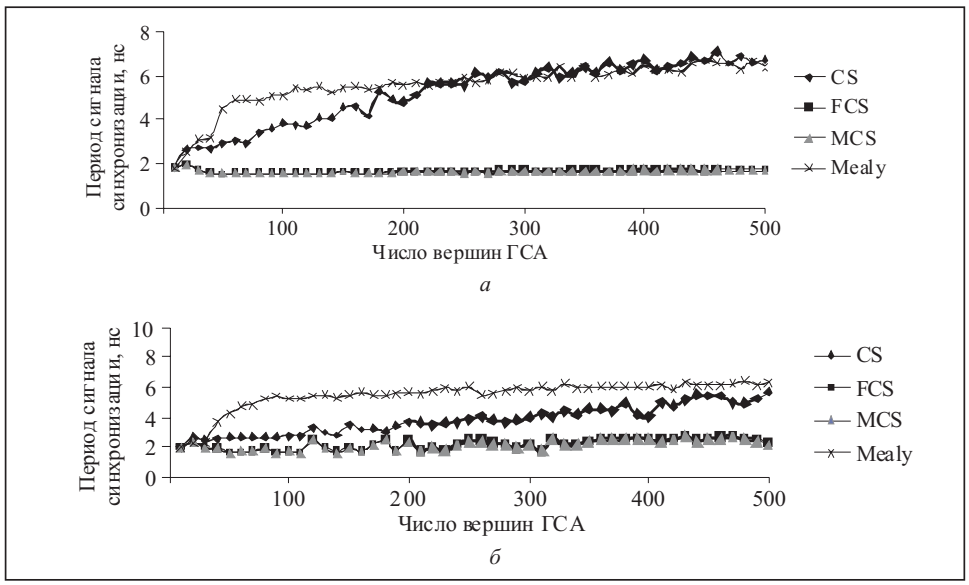

Рис. 5. Графики периода синхросигнала схем различных управляющих устройств, когда ГСА содержит 70 % операторных вершин (*а*) и 90 % операторных вершин (*б*)

Анализ графиков на рис. 5 позволяет сделать вывод о том, что предложенные структуры КМУУ не только занимают меньшую площадь, но и могут иметь меньшее время цикла, чем КМУУ с разделением кодов и автомат Мили. Однако сравнение периода сигнала синхронизации по-прежнему не позволяет сделать вывод о предпочтительной структуре управляющего устройства.

Проведенные исследования длительности формирования выходных функций (рис. 6) показали, что структура КМУУ с расширением формата микрокоманд формирует выходные функции быстрее КМУУ с модификацией ОЛЦ алгоритма управления для всех рассмотренных ГСА. Это обусловлено дополнительной схемой на выходе устройства, которая в моменты перехода автомата от одной ОЛЦ к другой не позволяет трактовать код класса ПОЛЦ как выходные сигналы.

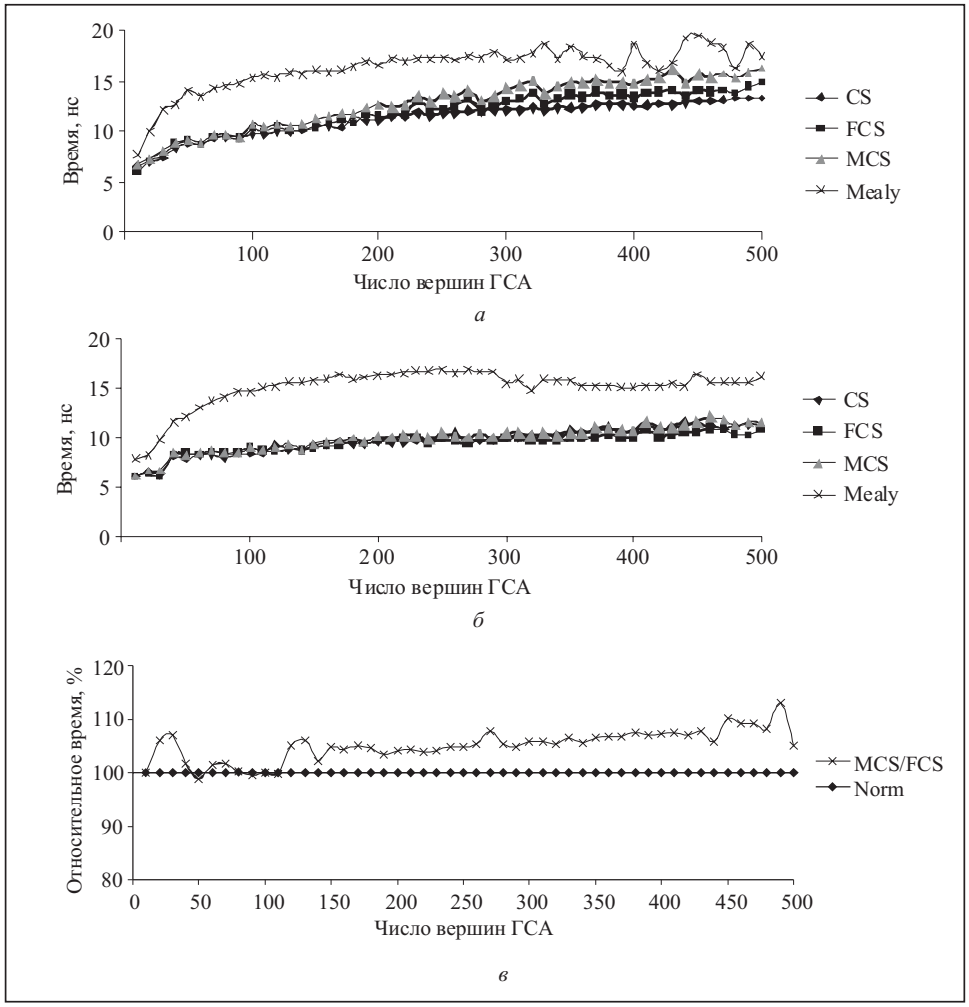

Рис. 6. Графики длительности формирования выходных функций УУ, когда ГСА содержит 70 % операторных вершин (*а*) и 90 % операторных вершин (*б*), а также графики относительной длительности формирования выходных функций, в случае, если ГСА содержит 90 % операторных вершин (значения для MCS-структуры нормированы относительно значений для FCS-структуры) (*в*)

Отметим, что все полученные схемы управляющих устройств с памятью использовали не более одного блока встроенной памяти микросхемы FPGA.

ЗАКЛЮЧЕНИЕ

Расширение формата микрокоманд и модификация ОЛЦ исходных ГСА показали снижение аппаратурных затрат в среднем на 40 % по сравнению с базовой структурой КМУУ при реализации схем УУ в базисе современных микросхем FPGA.

Для реализации предложенных структур КМУУ в базисе микросхемы Spartan-3 фирмы Xilinx авторами получена аналитическая зависимость между параметрами входной ГСА и необходимыми при реализации устройства аппаратурными затратами. Зависимость имеет вид $Q = (-0{,}026N^2 + 2{,}56N - 10{,}11)p$, где N — общее количество вершин ГСА, p — часть операторных вершин в ней, $p \in [0{,}5; 0{,}9]$. Данную зависимость можно использовать для других FPGA-микросхем фирмы Xilinx с четырехвходовыми LUT-элементами. Погрешность формулы для ГСА с количе-

ством вершин более 70 не превышает 20%, в то время как для малых ГСА статистическая погрешность может достигать величины аппаратурных затрат.

Длительность периода синхросигнала и время формирования выходных сигналов для предложенных структур КМУУ составляют приблизительно 1,7–2,5 нс по сравнению с 5–6 нс для автомата Мили, причем для КМУУ эти значения постоянны и зависят от типа микросхемы, а в автомате Мили задержки растут при увеличении сложности схемы. Использование специализированной САПР управляющих устройств многократно сокращает время разработки цифрового устройства (с нескольких часов до нескольких минут для ГСА небольшой сложности).

СПИСОК ЛИТЕРАТУРЫ

1. Baranov S. Logic and system design of digital systems. — Tallinn: TUT press, 2008. — 266 p.
2. Глушков В.М. Синтез цифровых автоматов. — М.: Физматгиз, 1962. — 476 с.
3. DeMicheli G. Synthesis and optimization of digital circuits. — N.Y.: McGraw-Hill, 1994. — 541 p.
4. Barkalov A., Titarenko L. Logic synthesis for compositional microprogram control units. — Berlin: Springer, 2008. — 272 p.
5. Баркалов А.А., Титаренко Л.А., Ефименко К.Н. Оптимизация схем композиционных микропрограммных устройств управления, реализуемых на ПЛИС // Кибернетика и системный анализ. — 2011. — № 1. — С. 179–188.
6. Грушвицкий Р., Мурсаев А., Угрюмов Е. Проектирование систем на микросхемах программируемой логики. — СПб: БХВ-Петербург, 2002. — 636 с.
7. Соловьев В.В., Климович А.С. Логическое проектирование цифровых систем на основе программируемых логических интегральных схем. — М.: Горячая Линия – Телеком, 2008 г. — 376 с.
8. Кузелин М. ПЛИС фирмы Xilinx: семейство SpartanTM-3. — http://chip-news.ru/archive/chipnews/200305/2.html
9. ROM-based FSM implementation using input multiplexing in FPGA devices / R. Senhadji-Navarro, I. Garcia-Vargas, G. Jiménez-Moreno, A. Civit-Ballcels // Electronics Letters. — 2004. — **40**, N 20. — P. 1249–1251.
10. Rawski M., Selvaraj H., Łuba T. An application of functional decomposition in ROM-based FSM implementation in FPGA devices // J. of Syst. Archit. — 2005. — **51**, N 6-7. — P. 424–434.
11. Sklyarov V. Synthesis and implementation of RAM-based finite state machines in FPGAs // 10th Intern. Conf. «Field-Programmable Logic and Applications: The Roadmap to Reconfigurable Computing», Proc., FPL 2000 Villach, Austria, Aug. 27–30, 2000. — P. 718–727.
12. Tiwari A., Tomko K.A. Saving power by mapping finite-state machines into embedded memory blocks in FPGAs // Proc. — Des., Automation and Test in Europe Conf. and Exhib. — 2004. — **2**. — P. 916–921.
13. Garcia E. Creating finite state machines using true dual-port fully synchronous selectRAM blocks // Xcell J. — 2000. — N 38. — P. 36–38.
14. Баркалов А.А., Титаренко Л.А., Цололо С.А. Оптимизация схемы автомата Мура, реализуемой в базисе ПЛИС // Кибернетика и системный анализ. — 2009. — № 5. — С. 180–186.
15. Barkalov A., Titarenko L. Logic synthesis for FSM-based control units. — Berlin: Springer, 2009. — 233 p.
16. Френкель Б.С., Кузьмич М.С. Xilinx WebPACK ISE. — http://ru.wikibooks.org/wiki/Xilinx_WebPACK_ISE.
17. Средства проектирования и программирования фирмы Altera. — http://www.altera.ru/cgi-bin/go?19.

Поступила 13.10.2011
После доработки 12.03.2012